Community-Based M2M Framework using Smart/HetNet Gateways for Internet of Things

Yi-Lan Lin Wu-Chun Chung Cheng-Hsin Hsu Yeh-Ching Chung

Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan
Email: {len08225, wcchung}@sslab.cs.nthu.edu.tw, {chsu, ychung}@cs.nthu.edu.tw

***Abstract*—In order to manage the Internet of things in a flexible and efficient way, this paper proposes a novel M2M framework using smart/HetNet gateways. Our approach is not only compatible to the M2M standard, but also enables the community-based coordination among gateways and devices. With smart and HetNet gateways, various types of requirements and applications can be fulfilled and handled at a local region. Accordingly, unnecessary network usage is avoided so as to reduce the traffic loads in mobile networks. We also implement a prototype to sustain an application scenario of IoT. In our prototype, the lamp is automatically switched on when a human face is detected. The demonstration shows that our system is practical and supports the subscription and notification in a community. Finally, experimental results reveal that some basic procedures can benefit from the shorter time and less uplink traffic if devices involved in an application are within the same region.**

Keywords. Machine to machine communications; internet of things; iot gateway; community-based; cloud computing

I. INTRODUCTION

As smart devices become a popular topic, more attentions are paid on making Internet of Things (IoT) or wearable devices be intelligent. Some smart devices can directly connect to the Internet and some legacy devices need to interact with an IoT gateway, e.g., a smart phone or a smart hub, for the Internet access. More applications might further require communications among IoT devices or between IoT devices and an IoT gateway. The device communications could be either wired or wireless connections. The wired connections could be over Ethernet, power line, and etc. and the wireless connections include Wi-Fi, Bluetooth, ZigBee, and etc. Due to the heterogeneity of connection technologies, how to efficiently leverage IoT devices is an important issue.

On the other hand, Machine to Machine (M2M) [1] applications are also getting more attentions for developing next generation mobile networks. A large amount of network communications coming from a huge growth of IoT devices will turn the mobile system into a bottleneck, e.g., resulting in high response time for IoT access. One solution is to upgrade the channel capacity or propose efficient scheduling and allocation algorithms for wireless communications to support massive network traffics. An alternative way is to fundamentally design a network stack for the coordination of large-scale IoT devices. For example, the core management is deployed on the cloud data center behind the mobile network while the M2M gateways are deployed on the edge to bridge the IoT devices. In such an environment, how these devices can communicate with each other and self-organize without human interactions are much important. Accordingly, providing a flexible framework is necessary to efficiently manage the IoT devices.

The European Telecommunications Standards Institute (ETSI) organization released some standards (i.e., smartM2M, oneM2M) [2] to define a set of message protocols for device registration to the gateway and also for information query or data acquirement from the gateways or the devices. OM2M [3] is an open source project and an ETSI-compliant M2M service platform. To support large amount of communications, OM2M exploits Publish and Subscribe (Pub/Sub), which is a loose-coupled scheme for data exchanges. Devices can subscribe data on the Service Capability Layer (SCL) rather than directly request the information from the target devices. When the new data is published from the target devices to the M2M gateway, M2M gateway notifies all devices which subscribed the information before. To avoid processing a large amount of requests, the Pub/Sub scheme provides an event-based interaction among all gateways and devices. IoT devices only need to publish data to M2M gateways or wait for the notification they have subscribed. Accordingly, M2M gateways play a major role to M2M applications.

To tackle aforementioned issues, this paper presents a comprehensive design from two aspects. Firstly, to elastically coordinate IoT devices for different types of applications, we proposes a novel framework based on OM2M and cloud computing. Secondly, to leverage various applications and heterogeneous wireless communications, we introduce two types of M2M gateways for our proposed framework. The main idea is to organize IoT devices into a region and group a set of regions into a community, instead of directly connecting all SCL to the Internet.

To implement a prototype system, we adopt Raspberry Pi, a single-board computer, to develop the gateway and the device. A heterogeneous network gateway (HetNet GW) is designed to connect devices with different types of connection technologies including Wi-Fi, Bluetooth, and Ethernet connections. We also adopt the minicomputer as a Smart GW to coordinate multiple HetNet GWs and connect to the Internet via mobile networks (e.g., 4G LTE). We further implement the redirecting mechanism based on

978-1-5090-0214-6/15 $31.00 © 2015 IEEE
DOI 10.1109/DSDIS.2015.19

OM2M for gateways to enable the registration with each other so that the messages can be exchanged from a region to another one. Some primitive procedures are also enhanced for devices to interact with the SCL, including registration, discovery, subscribe, and notification.

Our proposed framework adopts the Smart GW as a region master to manage devices within the same region and reduce the uplink traffic to mobile networks. On the other hand, the Smart GW supplies higher computing capability and larger storage space that ordinary IoT devices may not have. Our system is able to run locality-based computational work and store more data, e.g., streaming and processing video files. The experiments present the performance of basic procedures and the benefits to a concrete IoT application. The experiment results show that we can avoid unnecessary uplink traffics by offloading computing tasks of applications from cloud to the local gateways.

The rest of this paper is organized as follows. Section II reviews the related works while Section III presents the framework design and implementations. Section IV describes message flows and application scenarios based on our prototype. Experimental results are presented in Section V, and finally, conclusion remarks and future works are summarized in Section VI.

II. Related Work

M2M communications have been widely employed to data communications without or with limited human interventions among various devices [4]. These devices includes computers, embedded processors, smart sensors or actuators, and mobile devices, etc. Future M2M ecosystems will be complex and applied into many industries, including telecom and electronics [5]. The evolution of network architectures enables the mass deployment of M2M services, but the salient features of M2M traffic that may not be efficiently supported by current standards. Besides, M2M solutions fulfill very specific requirements that existing technologies are unable to complementary support. So, industry standards are required for M2M markets to sustain an explosive number of devices.

To solve a highly vertical fragmentation of current M2M markets and the lack of M2M standards, the ETSI released a set of specifications for a common M2M service platform [6]. The standard defines a functional architecture with a set of service capabilities for the M2M deployment on top of connectivity layers deployed in different domains. Each physical equipment of this architecture is represented as an instance of a SCL and a corresponding entity in the SCL is reachable by each device, gateways, and the network. The abstract model of each domain is a collection of Device SCL (DSCL), Gateway SCL (GSCL), and Network SCL (NSCL) [7]. Moreover, the resources of IoT applications are modeled as a hierarchical tree under each SCL instance which is responsible for corresponding device domain, gateway domain, and network domain.

OM2M is an ETSI-compliant M2M service platform which supplies the software to enable the function of SCL. OM2M provides RESTful APIs to enhance interoperations with SCL. A modular architecture is proposed running on top of an OSGi [8] layer, making it highly extensible via plugins. OM2M also enables the bindings of multiple communication protocols, the reuse of existing management mechanisms for remote devices, and the interworking with legacy devices.

To make the M2M system work, we need a M2M gateway connecting to IoT devices. This paper does not focus on improving the network performance of M2M gateway because there are no standard protocols for network communications among IoT devices. Actually, physical layer protocols may be interfered with each other and many works tried to overcome this issue. In [9], the authors proposes a heterogeneous IoT gateway based on dynamic priority scheduling algorithm to address the problem of data concurrency and improve the real-time performance. The proposed gateway realizes data conversion and transformation of RS485, Bluetooth, CAN, ZigBee, and GSM. The work in [10] proposes a novel service to allow real time interaction between mobile clients and smart/legacy things (sensors and actuators) via a wireless gateway. Furthermore, the M2M gateway in some scenarios acts as a data aggregator. Data aggregation is categorized into filtering, statistical calculation, and concatenation [11].

Different from prior works, this paper focuses on proposing a community-based coordination framework which is compatible to M2M architecture. Two kinds of gateways are also proposed to support different types of applications. The HetNet GW is able to leverage heterogeneous wireless communications while the Smart GW is equipped with more computing capability and storage space to consolidate other gateways and devices into a region. These gateways are all controllable and manageable from remote side. Moreover, the complicated application can be processed locally rather than push all data to the server for processing.

III. Design and Implementations

This section describes the design of a community-based coordination framework and how to integrate two types of our M2M gateways into the proposed framework.

A. Community-Based M2M Framework

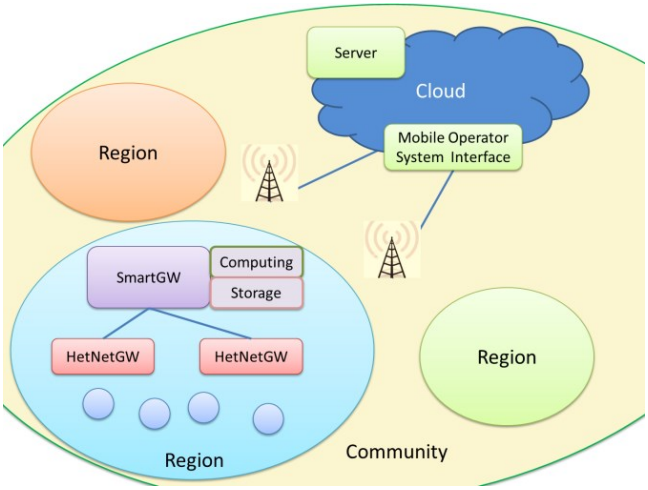

Fig. 1. Illustration of community-based framework

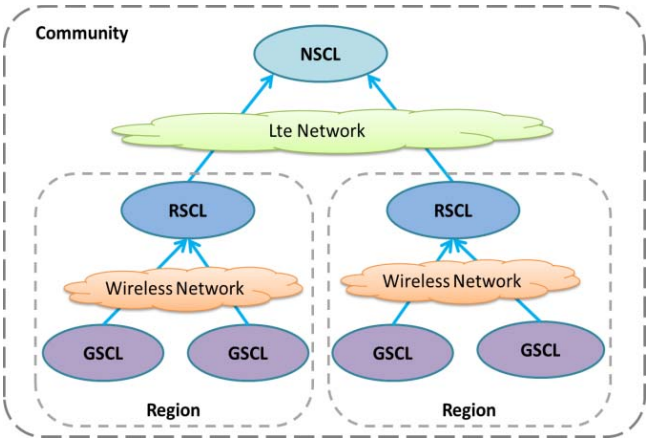

Fig. 2. Novel M2M framework with proposed Region SCL

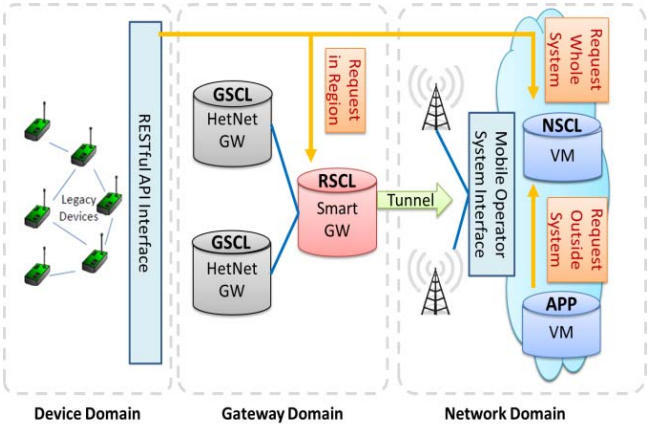

Fig. 3. Overview of prototype components for a single region

To prevent massive traffics transmitted from devices and gateways to the cloud servers, we design a two-tier framework to avoid unnecessary communication messages to the mobile network. Fig. 1 illustrates our framework. A community consists of multiple regions in which each region is composed of one Smart GW, a set of HetNet GWs are connected to the Smart GW, and a set of devices are connected to the HetNet GW. The division of community and region is pre-defined in this work. A region could be a place or a house. For a building, a region could be a lobby or a floor. In this case, the devices at different regions may need to communicate with each other. For example, a Camera device in the lobby region will notify the alarm in the office region. As a result, a community means an application running on cloud servers may need to consolidate IoT devices from different regions.

OM2M originally supplies NSCL for the network domain and GSCL for the gateway domain. The difference between these two SCLs is the relationship of registration, i.e., GSCL registers to NSCL. A set of GSCLs will be nodes of the resource tree of a NSCL. The application can use RESTful APIs to find resources from one NSCL to multiple GSCLs. OM2M provides a configuration file to specify the role of a NSCL or a GSCL with its corresponding NSCL. Messages among SCLs need a redirecting function for the forwarding. The latest released OM2M only supports to redirect a request either from NSCL to GSCL or from GSCL to its sibling GSCL. In our framework, a region is composed of one Smart GW and a set of HetNet GWs. We have to modify the redirecting function to meet our scenario.

Our novel framework is aligned with the hierarchy of NSCL and GSCL and try to extend the SCL in gateway domain. Fig. 2 illustrates an overview of our M2M framework. We define the SCL running on the edge of a region as Region SCL (RSCL). NSCL acts as a global service entrance on cloud. GSCL is running on HetNet GW and registers to RSCL running on Smart GW. Device in the same region is able to register to any HetNet GW within the wireless coverage. GSCL can communicate with RSCL via wireless networks within a region and RSCL communicates with NSCL via LTE networks within a community. Some applications such as smart home or smart classroom can be deployed on our Smart GW and not all data needs to be sent back to cloud servers.

An overview of our prototype components for a single region is shown in Fig. 3. In the network domain, the NSCL is hosting on a virtual machine (VM) in a cloud. The VM is associated with a public IP and acts as the entrance of M2M service. In the gateway domain, all Smart GWs are connected to NSCL over the mobile network. The IP address for Smart GW is allocated by the telecom system, which is usually not a public address on the Internet. Therefore, we exploit the tunnel technique for NSCL to connect the RSCL. The application server on cloud can query the information through NSCL and redirect the request to the gateway domain and finally retrieve the result from the device domain. For the device domain, we develop our semantic API for single-board devices to interact with SCL. When the application on the board actives, the service discovers all devices involved in this application. Devices may be located under the same gateway or in different gateways. M2M communications in the former cast can use the local IP address for notification messages. For the latter case, communications need GSCL or RSCL to further redirect the requests.

B. Smart/HetNet GW

This paper introduces two types of M2M gateways for a two-tier architecture. The HetNet GW is running with GSCL and acts as an ordinary M2M gateway, but not directly connecting to the Internet. In our framework, the HetNet GW is developed to leverage heterogeneous wireless communications and extend the service scale of a region for device management. We adopt the ARM-based single-board computer with embedded Linux OS to implement our HetNet GW. Both Wi-Fi and Bluetooth dongles are plugged to the HetNet GW. The HOSTAPD is compiled to act as an access point and the PAND is installed to act as Bluetooth NAP. The OpenvSwitch is further applied to bridge the WLAN interface and BNEP interface. We setup the DNSMASQ as the DNS and DHCP server. After Wi-Fi and Bluetooth are bound to the same bridge, we can assign IP addresses to these wireless interfaces in Linux. For the implementation of the device, WPA Supplicant is associated to Wi-Fi access point and PAND is paired to Bluetooth NAP on HetNet GW.

On the other hand, the Smart GW is running with RSCL to coordinate a set of HetNet GWs within a region and bridge

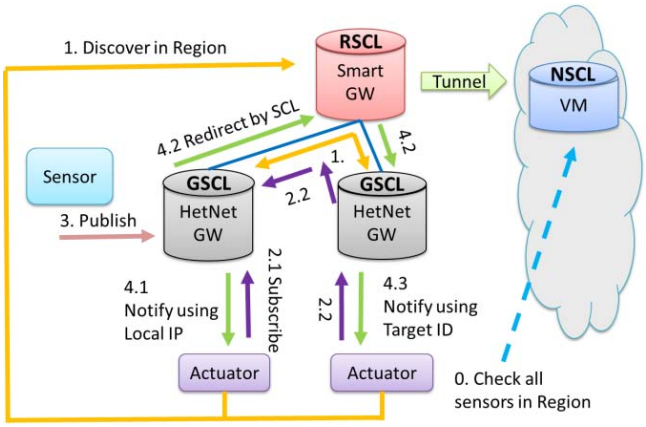

Fig. 4. Basic flows within a region

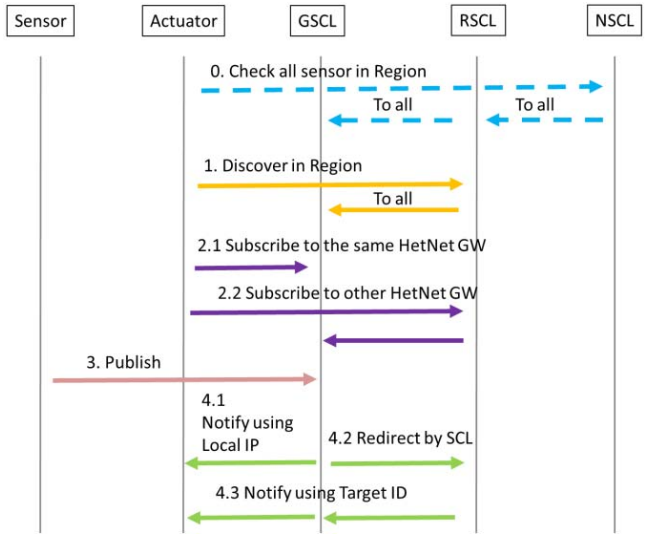

Fig. 5. Flowchart for basic flows within a region

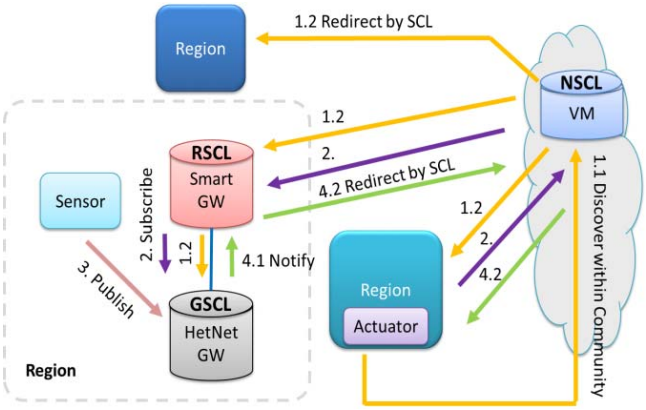

Fig. 6. Basic flows within a community

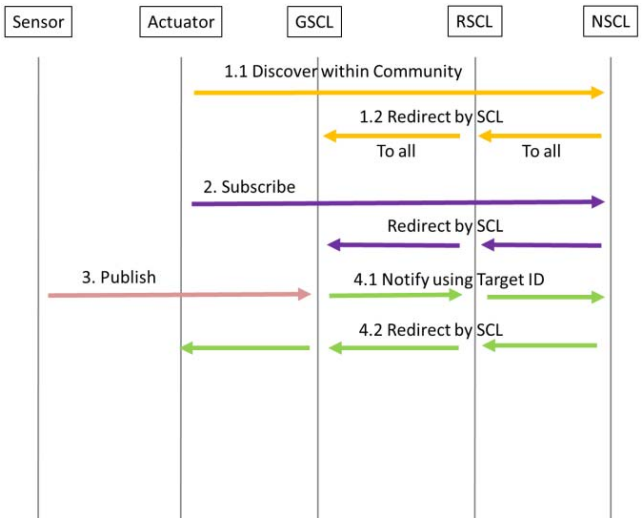

Fig. 7. Flowchart for basic flows within a community

the internal networking to the Internet via 4G LTE. The application running within a region is able to access RSCL on Smart GW rather than accessing the NSCL on cloud. We implement our Smart GW based on a minicomputer with commodity Intel Linux. Both Wi-Fi and 4G dongles are plugged to Smart GW, in which the 4G dongle is equipped with a configurable SIM card for LTE network connection. Since Smart GW has an advanced CPU and storage capability than HetNet GW, some complicated works or applications can be locally processing on Smart GW, e.g., face detection program and data aggregation.

With the design of Smart GW and HetNet GW, our prototype not only supports offloading computation from cloud to the local region, but also limits the forwarding messages within a region. As a result, the traffic load can be eliminated so as to save the bandwidth consumption for mobile networks.

IV. Message Flows and Operations

This section describes message flows for basic procedures within a region or within a community. Then, we introduce how to apply these operations into application scenarios.

A. Basic Flows and Operations

The basic flows include publish, subscribe and notify operations. When devices at the same region or in different regions, the flow operations of publish, subscribe, and notification are different. Different SCL domains handle the message flow in different ways, which will affect the uplink traffic to the mobile network.

1) Within a Regison

In some applications such as smart home, appliances in a house are set up and deployed within a region. If all devices are in the same region, the application can locate device resources from the RSCL. For the message forwarding, the device can directly use local IP address to connect other devices within the same GSCL; otherwise, GSCL redirects the message to other GSCL through RSCL.

We depict the basic flows for normal publish, subscribe, and notification operations within a region in Fig. 4. Fig. 5 presents the corresponding flowchart. Initially, the step 0, we need to check devices are within a region or not by requesting NSCL or preconfiguring in the application. After the actuator retrieves the location of desired devices, a discovery

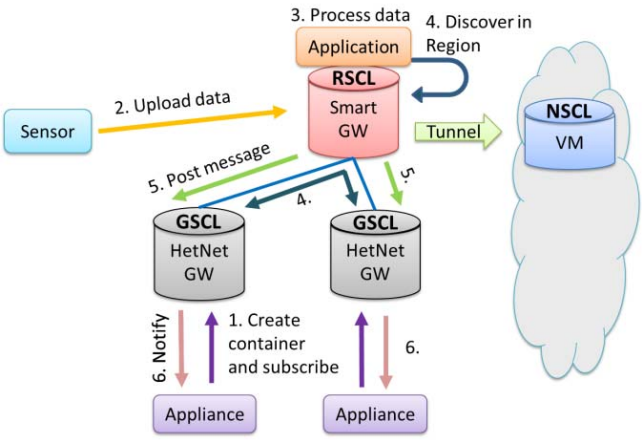

Fig. 8. Application flow on Smart GW within a region

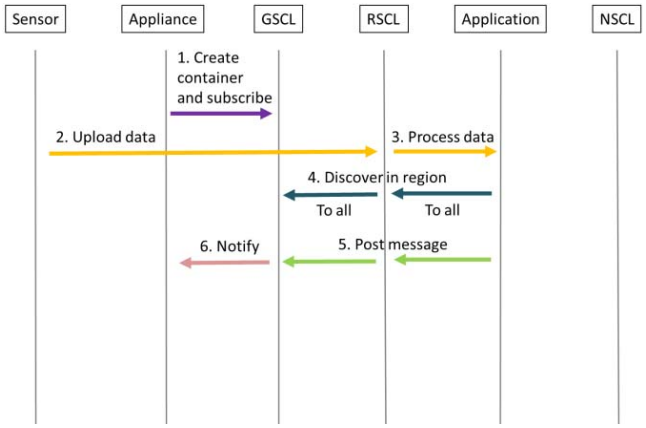

Fig. 9. Flowchart for application on Smart GW within a region

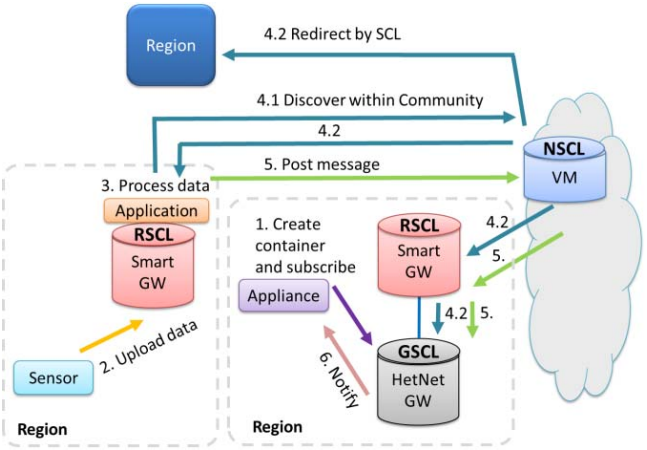

Fig. 10. Application flow on Smart GW within a community

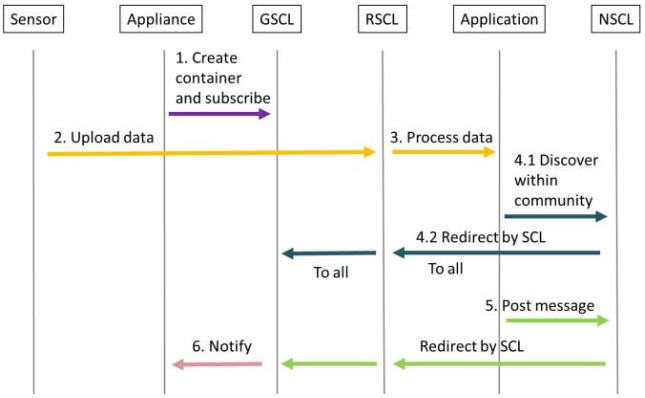

Fig. 11. Flowchart for application on Smart GW within a community

procedure is triggered to the RSCL. In step 1, the subscriber locates the publisher and issue the subscription to GSCLs in step 2. In step 3, the publisher posts new data to its GSCL periodically. Then, GSCL looks up the location of subscriber and sends the notification message. In step 4, if both publisher and subscriber are under the same HetNet GW, GSCL can easily send packets via local IP address. If the publisher and subscriber are located in different HetNet GWs, the notification message is forwarded by RSCL. In this case, we need to create a target ID for the subscriber, and then RSCL will redirect the message.

2) Within a Community

In some applications such as surveillance system, the camera and alarm devices may be distributed across multiple regions. When devices are located in multiple regions, the message has to be redirected by NSCL. That is, NSCL and RSCL will notify the devices according to the target ID.

Fig. 6 depicts basic flows when messages are redirecting within a community and Fig. 7 presents the corresponding flowchart. At first, in step 1, the subscriber issues the discovery message to NSCL and NSCL forwards the message to each region in order to locate the desired publisher. Then, the subscriber collects the device index of regions and subscribe the desired devices by filtering target IDs in step 2. After the publisher posts new date to its GSCL in step 3, the notification message is forwarded by SCL, and redirected by NSCL to another region in step 4. Since the communication between NSCL and RSCL is 4G LTE, the forwarding time across multiple regions will be affected by the network condition.

B. Application Flows and Operations

Originally, the data is uploaded to sever and computed by the application server. The application server then notifies the result to subscribers. In our framework, the application can be executed on Smart GW. This subsection describes the advanced flows with an application running on the gateway side.

1) Within a Region

When the desired devices and an application are located in the same region, all requests are forwarding by RSCL and GSCL within a region. In Fig. 8 and Fig. 9, we show the flows of publish, subscribe, and notify with the application on Smart GW. In step 1, appliances create and subscribe a container on GSCL to save data on the HetNet GW. When the sensor uploads data to Smart GW in step 2, the application will process the data in step 3 and discover all subscribers within the region in step 4. After filtering the container ID of

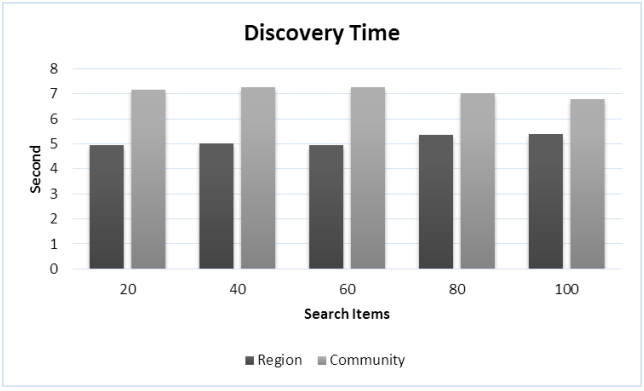

Fig. 12. Discovery time within a region and a community

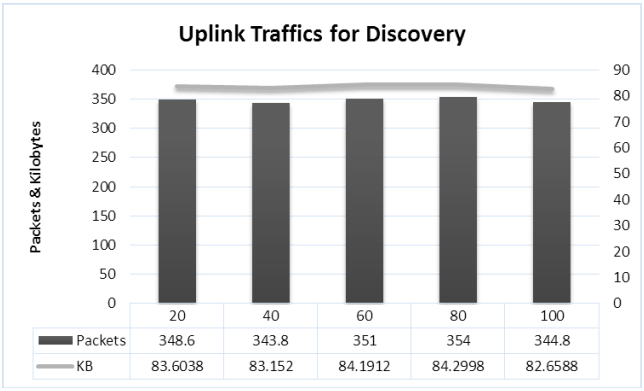

Fig. 13. Uplink traffics for discovery within a community

subscribers they ever issued the request, the application posts the result to corresponding containers on GSCL in step 5. Finally, GSCLs send the notification to appliances in step 6.

2) Within a Community

If the sensor and the appliance are located in different regions, the messages will be redirected by NSCL. In Fig. 10 and Fig. 11, we show the advanced flows when an application runs on Smart GW and the desired devices are located in different regions. In step 1, the appliance creates and subscribes a container on GSCL to save data on the HetNet GW. When the sensor uploads data to its Smart GW in step 2, the application preprocess the data in step 3 and discover the subscribers within the community in step 4. After filtering the container ID of all subscribers, the application posts the computing result to each subscribed container in step 5. Once GSCL receives the data on the container, GSCL sends the notification to appliance in step 6. This scenario shows that the devices may be deployed in multiple regions and the discovery procedure is necessary to look up all regions within the community. Thus, the uplink/downlink between RSCLs and NSCL will increase traffic loads to mobile networks.

V. Experiments and Results

This section presents a series of performance evaluations for the proposed community-based M2M framework. In experiments, we consider a community composed of two regions. The time of basic procedures are calculated for the request within a region and for the request within a community, including discovery time, subscription time, and notification time. Furthermore, we evaluate the realistic performance when applying our approach into a real scenario.

A. Testbed

Our prototype is set up from the network domain to the gateway and device domains. The cloud platform is built based on OpenStack [12]. A VM is launched on cloud to run the NSCL. The VM is equipped with single core CPU, 2 GB RAM, and 20 GB disk space. The operating system of the VM is Ubuntu 14.04. A public IP is associated with the NSCL VM for connecting M2M services over the Internet. The Smart GW is equipped with quad core Intel Core i3 CPU, 4 GB RAM, and 1 TB hard disk. The operating system is Ubuntu 15.04. The Smart GW connects to commercial LTE network operated by Taiwan Chunghwa Telecom with a HUAWEI E3276 4G dongle and also provides 802.11n Wi-Fi for communicating with HetNet GWs. The HetNet GW is developed based on a Raspberry Pi Model B+ and installed with the Raspbian operating system. The HetNet GW is equipped with 700 MHz single-core ARM1176JZF-S CPU, 512 MB RAM, and 16 GB SD card. Both 802.11n Wi-Fi and Bluetooth NAP are also supplied for heterogeneous wireless connections. The device is also a Raspberry Pi Model B+ and connects to HetNet GW via either Wi-Fi or Bluetooth. All the specs of each component are listed in Table 1.

Our testbed is composed of one NSCL, two RSCLs, and eight GSCLs. Two RSCLs running on different Smart GWs register to the NSCL VM on cloud and a half of GSCLs running on different HetNet GWs register to one RSCL. Network connections between NSCL and RSCL is 4G LTE while that between RSCL and GSCL or between GSCL and devices are wireless communications. More specifically, the Smart GW enables the Wi-Fi connection for HetNet GWs and the HetNet GW enables Wi-Fi or Bluetooth connections for devices. RSCL and GSCL have their local IP networks for the access in the same region.

B. Experimental Results

This subsection presents experimental results about the performance of basic procedures and the performance after applying a scenario.

1) Time for Discovery

Table 1. Spec of server, gateway, and device

	Server	Smart GW	HetNet GW	Device
CPU	1 core	4 core	1 core	1 core
Ram	2 GB	4 GB	512 MB	512 MB
Disk	20 GB	1 TB	16 GB	16 GB
Network	Wired	Wi-Fi, Bluetooth, 4G LTE	Wi-Fi, Bluetooth	Wi-Fi or Bluetooth

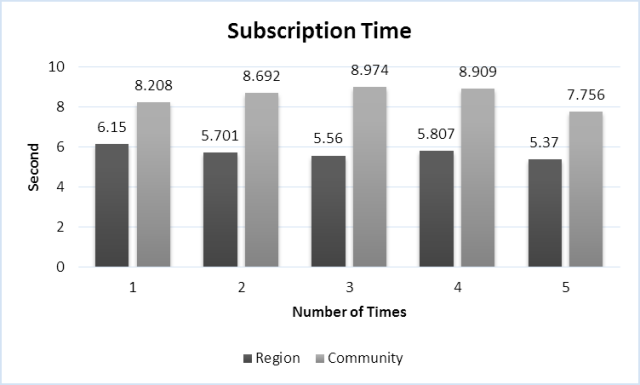

Fig. 14. Subscription time within a region and a community

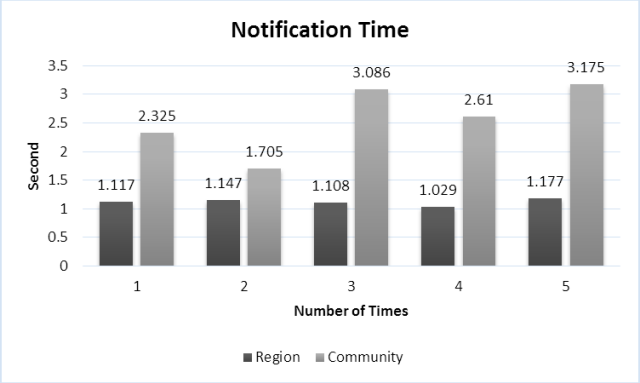

Fig. 15. Notification time within a region and a community

The discovery time for filtering a number of demanded devices is evaluated as follows. Since each device has its device ID, we generate 400 items of the device information distributed among eight GSCLs for the experiments. In Fig. 12, the number of desired devices is requested from 20 to 100. The discovery procedure will get the whole index of GSCL, so the number of devices does not affect the discovery time. However, the discovery time will be different if the discovery scale involves the same region or cross two regions of a community. Experimental results show that the discovery time is around 5 seconds and around 7 seconds when filtering the same set of devices within a region and within a community, respectively. Fig. 13 further depicts the uplink traffic of the discovery procedure within a community. The sum of total packets and the traffics consuming the LTE network are around 348 packets and 83 KB traffics.

2) Time for Subscription and Notification

The time of subscription and notification operations are evaluated respectively in a region and in a community. We sequentially test five times of each subscription and notification procedure. The subscription time is calculated by locating the devices and registering the subscription. The results shown in Fig. 14 depict that the improved performance is similar to the discovery time. The subscription time within a region is around 6 seconds and that time within a community is around 8 seconds. The performance of notification time is shown in Fig. 15 in which all the notification procedures

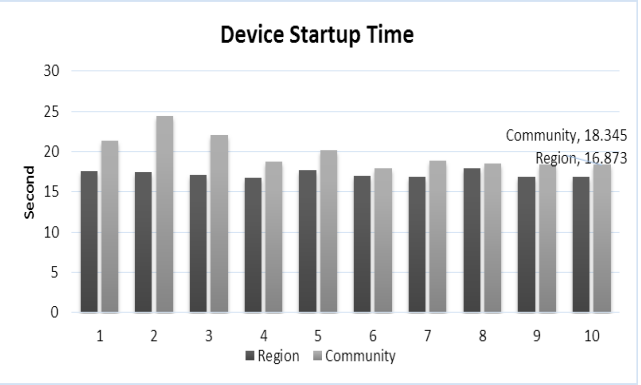

Fig. 16. Device startup time within a region and a community

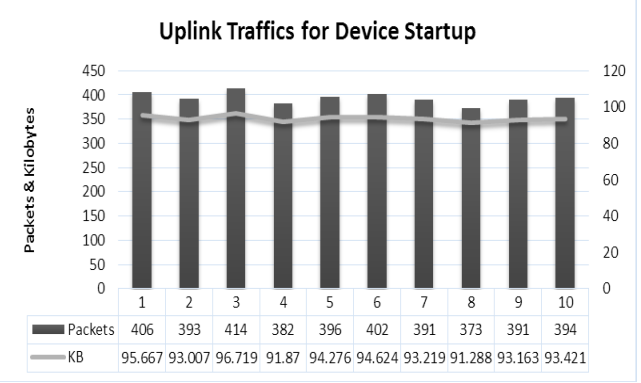

Fig. 17. Uplink traffics for device startup within a community

within a region takes around 1 second while that procedures within a community takes from 1.705 second to 3.175 seconds. Besides, the time within a region is smooth, but the time within a community is affected by the LTE network condition.

C. Application Scenario

To prove our approach is feasible, a scenario of smart home application is introduced and evaluated in this subsection. The appliance creates and monitors a container on gateway and the sensor uploads data to application on Smart GW. We deploy one device with the camera module on Raspberry Pi to act as an IP cam. The MJPG-streamer [13] is also adopted to stream the frames and save as a video file on the Smart GW. On another device, we use the GPIO to connect an LED to be a lamp. We implement the client software for turning on/off the lamp once receives the notification from SCL.

An application is developed on Smart GW to catch the streaming frames and detect a human face of each frame using OpenCV library. The LED device registers to a GSCL and waits for future notification from GSCL. Then, the application discovers the location of LED device and creates new data on corresponding GSCL if human faces were detected. When GSCL receives the new data, GSCL notifies the LED device to turn on the light.

Based on this scenario, we evaluate the performance of startup time for the LED device within a region and within a community. When the device starts up, the LED device firstly

discover whether other devices have the same device ID and remove them. Then, the LED device creates a new container for storing the data on GSCL and perform the subscription. In Fig. 16, the startup time of LED device takes around 16.873 seconds within a region and around 18.345 seconds within a community. The performance of uplink traffics for the scenario within a community is also presented. In Fig. 17, when the device startup is processed within a community, the traffic load of data communications passing through the LTE network is around 93 KB. We find that the discovery time affects the application startup and the uplink traffic is saved if the process is completed within a region.

VI. Conclusion and Future Works

In this paper, we introduce a novel community-based M2M framework. To efficiently manage devices, we further propose Smart GW and HetNet GW in a region. The Smart GW supplies more computing capabilities and larger storage spaces for coordinating IoT devices in a region or cross multiple regions of a community. Some complicated tasks and more data can be processed within a region so as to reduce the uplink traffic. Besides, our prototype can be deployed anywhere only if the mobile communications (e.g., 4G LTE) to a base station are available.

The experiments present preliminary results of basic procedures and application operations based in our prototype system. When devices involved in an application are all located in a region, the time of basic procedures can be improved. That is because our approaches reduce the needs of data transfers over mobile networks. If more regions are allocated in different locations, the variation of time will dramatically affect the performance of an application. Although the scales of regions and community are not large enough to represent future IoT environments, our proof-of-concept prototype system sheds some lights on opportunities of future development on the proposed framework.

In particular, this paper can be extended in several dimensions for the future works. First, deploying our prototype systems to more applications and more users will certainly lead us to many practical research problems. Second, the larger deployments will result in larger datasets from IoT devices. These datasets can be used to study cloud-based big data processing platforms. Last, more self-organized mechanisms for IoT device managements are critical for ubiquitous IoT deployments.

Acknowledgments

This work was partially supported by the Ministry of Science and Technology of Taiwan and the Industrial Technology Research Institute under grant numbers MOST 104-2221-E-007-053, MOST 104-3115-E-007-004, and ITRI 104A0058SB.

References

[1] T. Taleb, and A. Kunz, "Machine Type Communications in 3GPP Networks: Potential, Challenges, and Solutions," *IEEE Communications Magazine*, vol. 50, no. 3, 2012, pp. 178-184.

[2] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. S. Song, "Toward a Standardized Common M2M Service Layer Platform: Introduction to oneM2M," *IEEE Wireless Communications*, vol. 21, no. 3, 2014, pp. 20-26.

[3] OM2M - Open Source Platform for M2M Communication. http://www.eclipse.org/om2m/.

[4] M. Chen, J. Wan, S. Gonz´alez, X. Liao, and V. C.M. Leung, "A Survey of Recent Developments in Home M2M Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 98-114.

[5] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From Mobile to Embedded Internet," *IEEE Communications Magazine*, vol. 49, no. 4, 2011, pp. 36-43.

[6] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "OM2M: Extensible ETSI-compliant M2M Service Platform with Self-Configuration Capability," *Procedia Computer Science*, vol. 32, 2014, pp. 1079–1086.

[7] L. A. Grieco, M. B. Alaya, T. Monteil, and K. Drira, "Architecting Information Centric ETSI-M2M Systems," in: *IEEE International Conference on Pervasive Computing and Communications Workshops*, 2014, pp. 211-214.

[8] C.-L. Wu, C. -F. Liao, and L. C. Fu, "Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 2, 2007, pp. 193-205.

[9] D. Min, Z. Xiao, B. Sheng, H. Quanyong, and P. Xuwei, "Design and Implementation of Heterogeneous IOT Gateway Based on Dynamic Priority Scheduling Algorithm," *Transactions of the Institute of Measurement and Control*, vol. 36, no. 7, 2014, pp. 924-931.

[10] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT Gateway Centric Architecture to Provide Novel M2M Services," in: *IEEE World Forum on Internet of Things*, 2014, pp. 514-519.

[11] Y. Nakamura, A. Moriguchi, and T. Yamauchi, "CSDA: Rule-Based Complex Sensor Data Aggregation System for M2M Gateway," in: *International Conference Mobile Computing and Ubiquitous Networking*, 2015, pp. 108-113.

[12] OpenStack - Open Source Cloud Computing Software. https://www.openstack.org .

[13] P. Li and J.-P. Li, "Embedded Intelligent Home Control System Based on Arm-Linux," in: *International Conference on Wavelet Active Media Technology and Information Processing*, 2012, pp. 429-431.